Grid Configuration Issues

Paul Anderson
School of Informatics, University of Edinburgh

14th November 2003

Contents

1. Introduction ................................................................................................................ 1
2. Policy-driven Configuration ....................................................................................... 2
3. Distributed Policy Enforcement .................................................................................. 2
4. Federated Policy Specification ................................................................................... 3

This white paper is available online at
http://www.nesc.ac.uk/technical_papers/NeSC-2003-02.pdf

Copyright © 2003 The University of Edinburgh. All rights reserved.
1. **Introduction**

The problem of "configuration" occurs at several different levels in a typical Grid. For example, at the "Grid" level, applications must be configured to specify the permitted users and resources. At the "fabric" level, numerous services such as file services, and naming services, must be configured in detail to provide a functioning fabric on which to host the applications.

Most current configuration tools provide only a simple aid to configuration deployment, and the basic configuration parameters must be determined by a skilled manual process. The leads to many difficulties; for example, systems are unable to reconfigure autonomously to adapt in the case of failures, and it is difficult to guarantee the correctness the configurations.

The following paragraphs describe some specific research problems in the design of automatic configuration tools.

2. **Policy-driven Configuration**

In all cases, the aim of the configuration process is to create, and maintain, a system which ultimately conforms to some high-level policy specification. Given the current state of the technology, skilled administrators are required to interpret the (often informal) policy specification, and transform this into a set of detailed configuration parameters which yield a system with the required properties. It is often relatively easy to see that a particular configuration conforms to a specific policy, but existing tools generally provide only a small amount of assistance with the reverse process, and are incapable of generating configurations automatically to conform to specified policies.

3. **Distributed Policy Enforcement**

In a single system, or one with central control, it is possible to see how policies might be automatically transformed into appropriate configuration parameters. For example, the policy:

"There must always be two servers on this network"

could be interpreted by a tool which might select two (arbitrary) nodes on the specified network and configure them as servers. The controller could then monitor the servers and autonomically configure a replacement if one failed, thus maintaining conformance with the policy.

However, if there is no central control, then the set of eligible nodes might not even be known by any central controller; this situation is becoming more and more likely as systems use P2P models to deal with increasing scale and complexity. In most cases, it is
possible to conceive individual solutions using standard, or ad-hoc P2P protocols, but there is no generic method for handling general policy constraints. For example, a distributed application may want to specify:

"No more then 2Gb of disk space should be used in total (across all the participating nodes)"

It may not be desirable to have a central controller which is aware of all the participating nodes, and able to determine the disk space used by the application on each one.

4. Federated Policy Specification

In any sufficiently large system, different people will be responsible for different "aspects" of the system configuration. These people will want to specify policies that determine the behaviour of these aspects without being unnecessarily concerned with the other aspects. For example, a security expert might want to define policies on access control, without being concerned with the way in which database files are distributed across different physical disks, or the type of hardware. Note that the different aspects do not always have disjoint domains; a node may be both a "web server" and a "backup server", and both the of these aspects may place constraints of the disk usage, or the access control.

A configuration tool needs to be able to combine these policies to generate a configuration which satisfies their union. Current configuration languages often force users to over specify the policies which leads to conflicts between different aspects that cannot be resolved automatically. For example, if the language does allow "loose" specifications such as "use any server in this domain", then the user is forced to pick a specific arbitrary value which is likely to conflict with similarly arbitrary values specified in some other aspect.